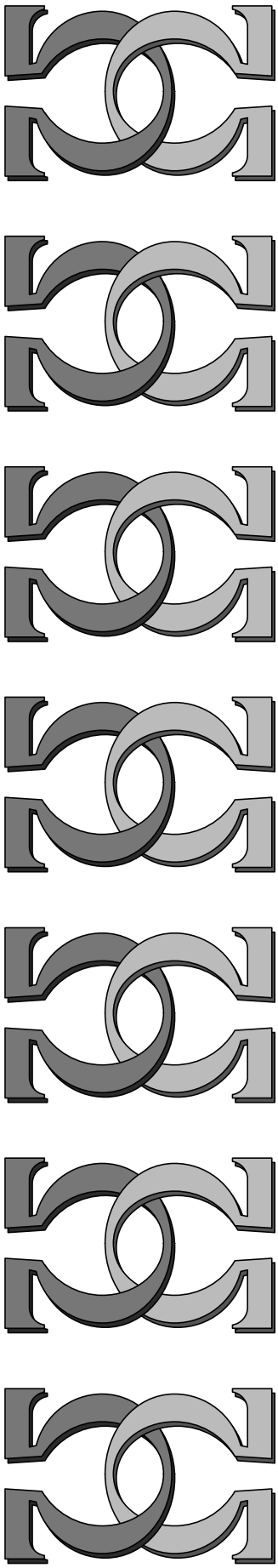


CDMTCS
Research
Report
Series

Automata with Equational Constraints

Michael J. Dinneen
Bakhadyr Khoussainov
Department of Computer Science
University of Auckland

CDMTCS-111
August 1999

Centre for Discrete Mathematics and
Theoretical Computer Science

Automata with Equational Constraints

Michael J. Dinneen and Bakhadyr Khoussainov

Department of Computer Science,
University of Auckland, Auckland, New Zealand
{mjd,bmk}@cs.auckland.ac.nz

Abstract

We introduce the concept of finite automata with algebraic constraints. We show that the languages accepted by these automata are closed under the Boolean operations. We give efficient polynomial-time algorithms for some decision problems related to these automata and their languages, including sufficient conditions for when we can determinize automata in polynomial time.

1 Introduction

The study of complexity-theoretic, algebraic or computability-theoretic properties of a set of problems is usually motivated by the fact that the set is closed under certain natural operations. For example, the set may be *Boolean*, that is closed under the operations of union, intersection and complementation. For instance, the set of all Turing decidable problems is Boolean. Also the set of problems decidable in polynomial time (e.g. the class P) and set of regular languages are Boolean classes. An important non-Boolean class, which forms a lattice under the union and intersection operations, is the set of all Turing recognizable problems. In general, the investigation of natural closure properties and their complexities for sets of languages is traditional for computer science.

In this paper we introduce and investigate new classes of decidable problems based on finite automata. Each of these classes is a subset of the regular languages. Informally each class consists of all languages accepted by finite automata whose transition tables satisfy a given set of algebraic constraints. We will show that these classes (of decidable problems) are Boolean which is of interest. Additionally, we will show that the set of all these classes forms a lattice. The introduction of these new classes allows us to recast standard problems in finite automata theory (e.g, the determinization problem, minimization problem) and solve new algorithmic questions about finite automata.

One motivation for introducing algebraic constraints comes from a purely computational point of view. Consider a set of tasks $\{a_1, a_2, \ldots, a_n\}$ to be executed on a computer. During a run the computer produces a sequence of states by executing the instructions for each task. Usually each run follows some set of rules specified by the

system software or constraints inherited from the system hardware. In a typical parallel environment the run may be allowed to follow different sequences of states to complete the set of tasks. The system may utilize the algebraic constraints of the type $a_i a_j = a_j a_i$ for some i and j. This allows execution of tasks a_i and a_j in any order (i.e., parallel execution).

Another motivation for studying automata with constraints comes from a purely algebraic point of view. Finite automata can be identified as finite algebras [2, 3]. Hence many fundamental concepts from algebra, in particular from the theory of finitely presented algebras (e.g., groups, semigroups) can be applied in the study of finite automata recognizable languages.

We now give a brief outline of the paper. In the next section we introduce automata with equational constraints, called E–automata, and provide two simple examples. We next study the properties of the languages accepted by E–automata, called E–languages, and show some closure properties of these languages. Also in Section 3, we study minimal deterministic E–automata and the structure between different classes of E–languages. Section 4 is devoted to the study of computational problems for E–automata. Here we discuss the issue of determinization of nondeterministic E–automata and provide sufficient conditions for when this can be done in polynomial time. We also present polynomial-time algorithms for solving some natural problems related to E–automata.

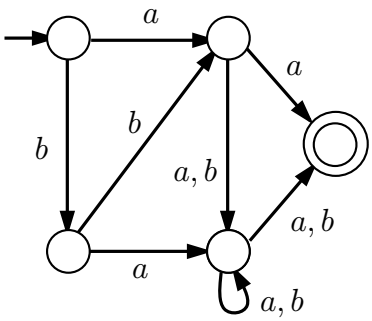
2 Preliminaries

We fix a finite alphabet Σ and let Σ^* denote the set of all finite strings over Σ. An *algebraic equation* is a pair of finite strings of Σ^*. We will use E to denote a fixed set of equations. Note that the set E may be infinite. Recall that a *finite automaton* M is a four tuple (S, I, Δ, F), where S is a finite set of states, I is the set of initial states, Δ is a transition partial function from $S \times \Sigma$ to subsets of S, and F is the set of final states. If for all $s \in S$ and $\sigma \in \Sigma$ the cardinality of $\Delta(s, \sigma)$ is one and I is a singleton then the automaton is *deterministic*. We can naturally extend the transition function Δ to the function Δ^* from from $S \times \Sigma^*$ to subsets of S. A string w is accepted by an automaton M if $|\Delta^*(s, w) \cap F| \geq 1$ for some initial state s in I.

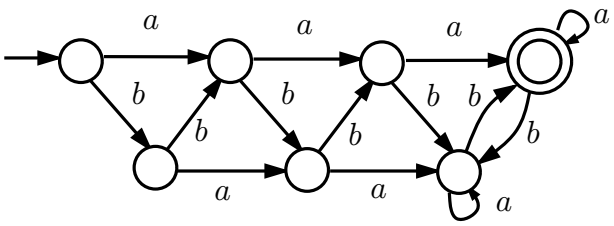
Definition 1. An *E–automaton* is a finite automaton (S, I, Δ, F) such that $\Delta^*(s, u) = \Delta^*(s, v)$ for all states $s \in S$ and equations (u, v) in E.

Note that any automaton satisfying some set E of equational constraints is considered an E–automaton. We now give two examples of E–automata.

Example 2. Let $E = \{(a_i a_j, a_j a_i) \mid i \neq j \text{ and } a_i, a_j \in \Sigma\}$. We call automata satisfying the equations E *commutative*. An example of a commutative automaton over $\Sigma = \{a, b\}$ is given below. The reader can verify that $\Delta^*(s, ab) = \Delta^*(s, ba)$ for all states s of the automaton.



Example 3. Let $E = \{(a, bb)\}$. An example of a deterministic E-automaton over $\Sigma = \{a, b\}$ is given below.



Note that the automaton in Example 2 is also an $\{(a, bb)\}$–automaton.

Definition 4. An *E–language* is the set of all strings in Σ^* that are accepted by some E–automaton.

We observe that every E–language is finite automata recognizable (i.e., a regular language). However, there exists regular languages that are not E–languages for some E.

We now want to characterize the E–languages of the previous two examples. First consider commutative automata. For a string w over $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ let the *signature* of w be the vector $(d_1, d_2, \ldots, d_n)$ where d_i is the number of σ_i occuring in w. Two strings are called *signature equivalent* if they have the same signature. A *signature-closed language* is a language L such that $w \in L$ implies $w' \in L$ for all w' signature equivalent to w.

Fact 5. Let L be a regular language. Then L is a signature-closed language if and only if it is recognized by a commutative automaton.

Proof. First assume L is a signature-closed language. Consider a minimum deterministic automaton M that accepts L. By the Myhill-Nerode Theorem, the states of M are $\sim$ equivalence classes where $w_1 \sim w_2$ if and only if for all $z \in \Sigma^*$, $w_1 \cdot z \in L \iff w_2 \cdot z \in L$. We show that M is a commutative automaton. It suffices to show that if two strings w_1 and w_2 have the same signature then $w_1 \sim w_2$. Suppose $w_1 \not\sim w_2$. Then there exists a z such that, without loss of generality, $w_1 \cdot z \in L$ and $w_2 \cdot z \notin L$. If w_1 and w_2 are signature equivalent then also $w_1 \cdot z$ and $w_2 \cdot z$ are signature equivalent. Since L is signature-closed $w_1 \cdot z \in L$ implies $w_2 \cdot z \in L$. This is a contradiction. Hence $\Delta^*(s, ab) = \Delta^*(s, ba)$ for all states s of M, where Δ is the transition function of M.

Now assume L is accepted by a commutative automaton M. If $w = w_1 w_2 \ldots w_n$ is accepted by M than so is any permutation w' of w. Note that w and w' have the same signature. Thus, L is a signature-closed language. □

Regarding our second example above say that two strings w_1 and w_2 are (a, bb)–*equivalent* if w_2 can be obtained from w_1 by a finite sequence of substring replacements a with bb or bb with a. A language L is said to be (a, bb)-*closed* if $w \in L$ implies $w' \in L$ for each w' that is (a, bb)–equivalent to w. One can prove the following result.

Fact 6. Let L be a regular language. Then L is (a, bb)-closed if and only if it is recognized by an E–automaton, where $\{(a, bb)\} \subseteq E$.

3 Properties of E–Languages

We will present the basic properties of E–languages in three parts. We first show that the class of E–languages for fixed E is closed under the Boolean operators. We next show that the minimization of a deterministic E–automaton preserves the equations from E. In the third part we compare the classes of E–languages and show that the set of all classes of E–languages forms a lattice.

3.1 Set-theoretic properties

In this subsection we study set-theoretic properties of E–languages. Let the alphabet Σ and the set of equations E be fixed throughout this section. We first show that E–languages are closed under union and intersection.

Lemma 7. If L_1 and L_2 are E–languages then so are $L_1 \cup L_2$ and $L_1 \cap L_2$.

Proof. The proof follows the standard constructions for automata. Let $A_1 = (S_1, I_1, \Delta_1, F_1)$ and $A_2 = (S_2, I_2, \Delta_2, F_2)$ be E–automata accepting the languages L_1 and L_2, respectively. Assume that S_1 and S_2 have no states in common. The automaton $(S_1 \cup S_2, I_1 \cup I_2, \Delta_1 \cup \Delta_2, F_1 \cup F_2)$ accepts the union $L_1 \cup L_2$. One also observes that this is an E–automaton.

We now construct an E–automaton that accepts $L_1 \cap L_2$. Consider the automaton $(S_1 \times S_2, I_1 \times I_2, \Delta, F_1 \times F_2)$ where

$$\Delta((s_1, s_2), \sigma) = \Delta_1(s_1, \sigma) \times \Delta_2(s_2, \sigma), \text{ for all } s_1 \in S_1, s_2 \in S_2, \text{ and } \sigma \in \Sigma.$$

Note that this automaton accepts the intersection of the two languages. To show that this automata is an E–automaton take $(u, v) \in E$ and $(s_1, s_2) \in S_1 \times S_2$. Then the following equalities hold:

$$\Delta^*((s_1, s_2), u) = \Delta_1^*(s_1, u) \times \Delta_2^*(s_2, u) = \Delta_1^*(s_1, v) \times \Delta_2^*(s_2, v) = \Delta^*((s_1, s_2), v).$$

This shows that the constructed automaton is an E–automaton. □

To show that E–languages are closed under complementation we need a method to convert E–automata to deterministic E–automata. Again, we will show that the standard subset construction preserves equations from E.

Recall that the determinization of an automaton $M = (S, I, \Delta, F)$ is an automaton $M_d = (S_d, I_d, \Delta_d, F_d)$ where

$$\begin{array}{rcl} S_d & = & \{X \mid X \subseteq S\} \\ I_d & = & \{I\} \\ \Delta_d(X, \sigma) & = & \bigcup_{x \in X} \Delta(x, \sigma), \text{ for } X \in S_d \text{ and } \sigma \in \Sigma \\ F_d & = & \{X \mid X \in S_d \text{ and } |X \cap F| \geq 1\}. \end{array}$$

We note that M and M_d accept the same language.

Lemma 8. If M is an E–automaton then so is M_d.

Proof. It suffices to check that M_d satisfies the equational constraints of E. Indeed, for $(u, v) \in E$ and $X \in S_d$ we have

$$\Delta_d^*(X, u) = \bigcup_{x \in X} \Delta^*(x, u) = \bigcup_{x \in X} \Delta^*(x, v) = \Delta_d^*(X, v).$$

□

We can use the previous lemma to show that E–languages are closed under complementation.

Lemma 9. If L is an E–language then so is the complement $\overline{L} = \Sigma^* \setminus L$.

Proof. Since L is an E–language there exists an E–automaton M that accepts L. By the previous lemma we can construct a deterministic E–automaton $M_d = (S, I, \Delta, F)$ that also accepts L. The automaton $(S, I, \Delta, S \setminus F)$ is an E–automaton and accepts $\overline{L}$. □

We now mention that some standard closure properties of regular languages do not hold for E–languages. Consider the concatenation of two languages $L_1 \cdot L_2 = \{u \cdot v \mid u \in L_1 \text{ and } v \in L_2\}$. Take $L_1 = \{ab, ba\}$ and $L_2 = \{a\}$, which are both signature-closed languages. However, the concatenation language $L_1 \cdot L_2 = \{aba, baa\}$ is not a signature-closed language.

3.2 Minimal deterministic E–automata

We show that the standard procedure for minimizing deterministic E–automata preserves the set of equations E. Recall that an automaton is minimal if it has the fewest number of states for accepting a given regular language. For a regular language L there is an unique minimal deterministic automaton M_L [4].

Theorem 10. If L is an E–language then M_L is an E–automaton.

Proof. We use the Myhill-Nerode Theorem. Take a deterministic E-automaton $M = (S, I, \Delta, F)$ that accepts L. Assume all states of M are reachable from the initial state.

The states of the minimal automaton $M_L = (S_L, I_L, \Delta_L, F_L)$ correspond to the equivalence classes of the following relation. Two states s_1 and s_2 of M are equivalent, denoted by $s_1 \sim s_2$, if for all $u \in \Sigma^*$,

$$\Delta^*(s_1, u) \in F \iff \Delta^*(s_2, u) \in F.$$

Recall that F_L is the equivalence classes that contain states of F. Δ_L is defined as follows. For all $s_L \in S_L$ and $\sigma \in \Sigma$, $\Delta_L(s_L, \sigma) = \Delta(s, \sigma)_L$, where s_L is the equivalence class containing the state s.

Now we show that M_L is an E–automaton. Indeed, let $s_L \in S_L$ and $(u, v) \in E$. Then

$$\Delta_L^*(s_L, u) = \Delta^*(s, u)_L = \Delta^*(s, v)_L = \Delta_L^*(s_L, v).$$

□

3.3 Global structure of E–languages

The previous sections indicate that we can study the class of E–languages for a given set of equations E. For example, E–languages are closed under the union, intersection and complementation operations. Many other results for regular languages also hold for E–languages and lead to the study of their computational properties.

The introduction of E–languages gives rise to a natural algebraic structure that relates classes of E–languages, for different E. The aim of this subsection is to introduce and investigate properties of this structure.

We define the class $R_E = \{L \mid L \text{ is an } E\text{–language}\}$ of regular languages for a given alphabet Σ and set of equations E. For example, $R_\emptyset$ is the set of all regular languages. Also when $E = \{\sigma = \lambda \mid \sigma \in \Sigma\}$, where λ is the empty string, $R_E = \{\emptyset, \Sigma^*\}$.

Consider the set $\mathcal{C} = \{R_E \mid E \subseteq \Sigma^* \times \Sigma^*\}$ of classes of E–languages. Informally, each element of this set represents the class of languages that are decidable by automata that satisfy certain algebraic constraints. Naturally this set is partially ordered under the set-theoretic inclusion. Note that for $E_1 \subseteq E_2$ we have $R_{E_2} \subseteq R_{E_1}$. The following lemma shows that this partially ordered set is also a complete lower semi-lattice, that is every subset of $\mathcal{C}$ has a greatest lower bound.

Lemma 11. Let $\{E_i\}$ be a set of equations and let $E = \bigcup E_i$. Then $R_E = \bigcap R_{E_i}$.

Proof. If L is an E–language then L is also an E_i–language for any i. Hence $R_E \subseteq \bigcap R_{E_i}$. Assume that L is an E_i–language for every i. Take a deterministic E_i–automaton M_i that accepts L for each i. Furthermore, by Theorem 10, we can assume each M_i is minimal. Since there is an unique deterministic minimal automaton M for L we conclude that $L \in R_E$. Therefore $R_E = \bigcap R_{E_i}$. □

We can also show that $\mathcal{C}$ is an upper semi-lattice.

Corollary 12. Let X be a class of regular languages. Then there exists a minimum class $R_{E'}$ that contains X. Hence for any $R_{E_1}, R_{E_2} \in \mathcal{C}$ there exists a least upper bound containing both R_{E_1} and R_{E_2}.

Proof. Let $I = \{E \mid X \subseteq R_E\}$. Note that $I \neq \emptyset$ because $X \subseteq R_\emptyset$. By the previous fact $\bigcap_{E \in I} R_E = R_{E'}$ were $E' = \bigcup_{E \in I} E$. Clearly, $R_{E'}$ is the desired class of languages. □

We can combine the above two results into the following theorem.

Theorem 13. The set $\mathcal{C} = \{R_E \mid E \subseteq \Sigma^* \times \Sigma^*\}$ of classes of E–languages forms a lattice $(\mathcal{C}, \vee, \wedge)$, where $R_{E_1} \vee R_{E_2}$ corresponds to taking the minimum R_E that contains $R_{E_1} \cup R_{E_2}$ and $R_{E_1} \wedge R_{E_2}$ is equal to the intersection $R_{E_1} \cap R_{E_2}$.

4 Algorithmic Problems for E–Automata

In this section we study computational issues regarding to the determinization process of E–automata and related decision problems.

4.1 Determinization of E–automata

We now recall the standard process for generating the states of a deterministic automaton M_d from a nondeterministic automaton $M = (S, I, \Delta, F)$. The states of M_d are subsets of S. This process works in levels where

$$L_1 = \{I\} \text{ and } L_{i+1} = \bigcup_{\sigma \in \Sigma} \{\bigcup_{t \in T} \Delta(t, \sigma) \mid T \in L_i\} \setminus (L_1 \cup L_2 \cup \ldots \cup L_i).$$

This generation process terminates at *height* k where $L_k = \emptyset$. Define $height(M) = k$. Below we will consider the height function restricted to a class $\mathcal{K}$ of automata. The class $\mathcal{K}$ has *polynomial height* if there exists a polynomial p such that $height(M) \leq p(|M|)$ where $|M| = |S|$ for all $M \in \mathcal{K}$.

For an automaton M the determinization process may yield a deterministic automaton M' that is exponential in the size of M, that is, one state M' for each subset of states of M. For example, the language

$$L = \{w \in \{0,1\}^* \mid w \text{ has a 1 in the } n\text{-th character from right}\}$$

has a (nondeterministic) automata with $n + 1$ states. One can see that the minimal deterministic automaton that accepts L requires at least 2^{n-1} states. Below we show that for some types of E–automata we can bound the number of states for the determinized automata by a polynomial function of n, where n is the number of states of the input automaton.

Definition 14. Fix a set of equations E. Two strings u and v are *E–equivalent* if for any E–automaton $M = (S, I, \Delta, F)$, $\Delta^*(s, u) = \Delta^*(s, v)$ for all $s \in S$. The *breadth function* $b_E(n)$ is defined as the number of E–non-equivalent strings of length n.

Intuitively the breadth function gives us an upper bound on the maximum number of states generated (at level n) during a breadth-first determinization of an E-automaton. We now give an example.

Example 15. Let $E = \{(a_i a_j, a_j a_i) \mid i \neq j \text{ and } a_i, a_j \in \Sigma\}$. Recall that E–automata are commutative. With $k = |\Sigma|$ the breadth function $b_E(n) = \binom{n+1}{k-1} = O(n^{k-1})$.

We now present sufficient conditions for when the determinization process runs in polynomial time.

Theorem 16. Let $\mathcal{K}$ be a class of E–automata that satisfies the following two conditions:

1. the class $\mathcal{K}$ has polynomial height, and
2. the breadth function b_E is bounded by a polynomial.

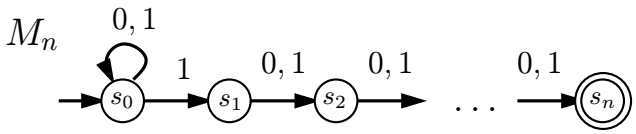
Then we can determinize any $M \in \mathcal{K}$ in polynomial time.

Proof. The determinization process will generate exactly $height(M)$ levels in computing the states of the deterministic equivalent automaton M_d. At level i there are at most $b_E(i)$ states included in set L_i. Since Σ is fixed, generating level $i+1$ from level i takes time proportional to the size of L_i. We can check in constant time, using an appropriate dictionary data structure (e.g. virtual array [1]), whether a state X (subset of S) of the deterministic machine M_d has been previously generated. We can generate X in time proportional to the size of M. Since M is in a class $\mathcal{K}$ of polynomial height, the number of states of the deterministic machine is at most $b_E(height(M)) \cdot height(M)$. Thus, the total time to produce the deterministic machine is polynomial bounded. □

The above theorem is strong in the sense that we need both hypothesis. We give two observations to illustrate this.

Fact 17. There exists a class of automata $\mathcal{K}$ that has polynomial height but for which the determinization process requires exponential time.

Proof. For $E = \emptyset$ and $\Sigma = \{0, 1\}$ consider the family $\mathcal{K} = \{M_n\}$ of automata where M_n is displayed below.



The automaton M_n accepts the previously mentioned language

$$\{w \in \{0,1\}^* \mid w \text{ has a 1 in the } n\text{-th character from right}\}.$$

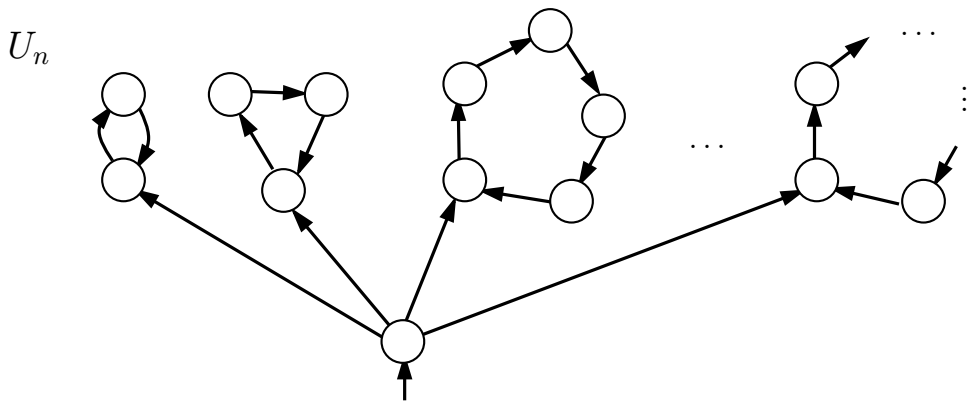
We claim that $height(M_n)$ is $n+1$. Take any generated state $X = \{s_{i_1}, s_{i_2}, \ldots, s_{i_k}\} \subseteq \{s_0, \ldots, s_n\}$ where $i_j < i_{j+1}$. We show that there is a string of length at most $n+1$ that yields X in the determinization process. Note that $s_{i_1} = s_0$ for all possible X. If $k = 1$ then X is $\{s_0\}$ which is the initial state $\{I\}$. Otherwise $k > 1$ and denote 0^0 as an empty string. Now consider the string

$$w = 0^{(i_k - i_{k-1} - 1)} 1 \ldots 0^{(i_3 - i_2 - 1)} 1 0^{(i_2 - i_1 - 1)} 1$$

The reader can easily see that the state X is generated at level $|w|$ which is at most $n+1$. Hence $\mathcal{K}$ has polynomial height. Clearly there are exponential possible X so the proof is completed. □

Fact 18. There exists a class of E–automata $\mathcal{K}$ such that the breadth function b_E is bounded by a polynomial but for which the determinization process requires more than polynomial time.

Proof. In this example let $E = \emptyset$ and $\Sigma = \{0\}$. Clearly for any unary alphabet, the breadth function $b_E(n) = 1$ is bounded by a polynomial. Consider the family $\mathcal{K} = \{U_n\}$ of automata where U_n is displayed below.

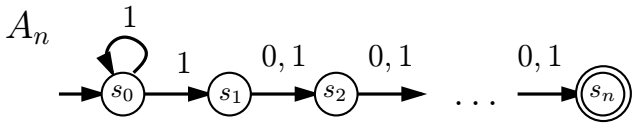


The automaton U_n has n disjoint cycles where the cycle i from the left is of length p_i, where p_i is the i-th prime number. Besides the singleton start state, the subsets of states generated by the determinization process all have cardinality n. In fact, any combination of states from the cycles (one from each) will be generated since the lengths of all the cycles are relatively prime. Note that there are exponential number of possibilities so the proof is completed. □

We mention that the class of E–languages over an unary alphabet, where E is not trivial, is finite. Therefore the case $E = \emptyset$ is required for the previous fact.

We now give one example of an application of Theorem 16.

Example 19. Let $E = \{00 = 01\}$ and $\Sigma = \{0, 1\}$. The breadth function $b_E(n) = n+1$ since the strings $0^n, 10^{n-1}, 1^20^{n-2}, \ldots, 1^n$ are representatives of the E–equivalence classes of strings of length n. Consider the family $\mathcal{K} = \{A_n\}$ of automata where A_n is displayed below



Note that each A_n is an E–automaton. The automaton A_n accepts the language

$$L_n = \{1^m \cdot w \mid m \in \{1, 2, \ldots\} \text{ and } w \in \Sigma^n\}.$$

The class $\mathcal{K}$ has polynomial height by an argument similar to the one given for the automaton M_n in Fact 17. By the above theorem the determinization process for class $\mathcal{K}$ yields deterministic automata in polynomial time (and polynomial size) of A_n. In fact, the determinization is quadratic and moreover yields the minimum deterministic automaton for L_n.

4.2 Recognizing E–automata and E–languages

Several problems occur naturally when dealing with E–automata. In this section we address a few of them for a fixed finite set E.

Theorem 20. For a given automaton M we can decide in time $O(n \cdot m)$ whether M is an E–automaton, where n is the number of states of M and m is the number of transitions of M. Furthermore, if M is deterministic we can decide in linear time whether M is an E–automaton.

Proof. Let S be the set of states of M. For a given state s of S we can determine whether s satisfies E by checking if $\Delta^*(s,u) = \Delta^*(s,v)$ for all $(u,v) \in E$. First note that the lengths of u and v are independent of n. To calculate $\Delta^*(s,u)$ we build a sequence of subsets $S_0 = \{s\}, S_1, S_2, \ldots, S_{|u|}$ of S. For the i-th character σ in u we generate $S_i = \bigcup_{x \in S_{i-1}} \Delta(x, \sigma)$ This union can be done in time $O(m)$, so determining $\Delta^*(s,u)$ is at most $O(|u| \cdot m) = O(m)$. Now since the number of equations is finite we can decide, in time $O(m)$, whether a given state s satisfies E. After repeating this check for all $s \in M$, we see that the total running time is $O(n \cdot m)$.

If M is a deterministic automaton then $\Delta^*(s,u)$ is a single state and can be computed in time proportional to the length of u, which is a constant. So we can decide whether a given state s satisfies E in constant time. Thus, after checking all states, we can decide if M is an E–automaton in time $O(n)$. □

Corollary 21. For a given language L accepted by a deterministic automaton M we can decide in time $O(n^2)$ whether L is an E–language, where n is the number of states of M.

Proof. By standard techniques we can minimize the deterministic automaton M in $O(n^2)$ time [4]. Now the constructed minimal automaton M_L accepting L is an E–automaton by Theorem 10 if and only if L is an E–language. By the above theorem we can decide whether M_L is an E–automaton in linear time. Thus whether L is an E–language is decided in $O(n^2)$ time. □

We observe that, since any regular language is accepted by a deterministic automaton, we can decide whether any regular language is also an E–language.

5 Conclusion

This paper develops the new notion of E–automata and studies the languages that they accept. Emphasis was given to show that these E–languages correspond to natural

classes of decision problems within the regular languages. We showed that we can decide whether an automaton is an E–automata in polynomial time (linear time for deterministic automaton). We have also shown that knowing what algebraic constraints a family of automata satisfies may be useful in predicting the growth of the determinization process. We think that the concept of automata with equational constraints is useful from a computational and algebraic point of view. We hope that research in this direction may lead to interesting results in the theory of formal languages and automata.

References

[1] G. Brassard and P. Bratley. *Fundamentals of Algorithms* (Chapter 5), Prentice Hall, 1996.

[2] J. R. Büchi. *Finite Automata, Their Algebras and Grammars: Towards a Theory of Formal Expressions*, D. Siefkes (editor), Springer-Verlag, 1989.

[3] F. Géceg and M. Steinby, *Tree Automata*, Akadémiai Kiadó, Budapest, 1984.

[4] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*, Addison–Wesley, 1979.